CD-1 Git Lab Level 1

Objectives

1. In this lab students will practice using the following git commands: `init`, `clone`, `status`, `add`, and `commit`.
2. At the end of this lab students should be comfortable making simple changes to files in a git repository.

Overview

You have already learned about the basic commands in git and how to use them. Here is a brief summary of those commands:

- `git init` -- initialize a new git repository in the current directory
- `git clone` -- create a local copy of a git repository located somewhere else (called a remote repository)
- `git status` -- display the current status of the git repository, including files that have been changed or that are not tracked in git yet
- `git add` -- stage the selected file(s) to be added the git repository
- `git commit` -- commit staged changes to your local repository

Unit Test

For this lab your submission to GitHub will be checked to confirm that you changed one file by adding your user name to it.

Instructions

In this lab you will practice using the basic git commands listed above. In part one you will create a new git repository and add a file to it. In part two you will clone a remote repository, change a file, and submit your changes.

Part 1

- Create a new directory called "GitLab01" and navigate to it in your terminal.
- In GitLab01, create another new directory called myRepo

- In MyRepo, initialize a new git repository
- Create a new file and add it to the repository you just initialized Hint, you can create files in terminal with `touch myfile`
- Check the status of your repo to confirm that it is up to date with no pending changes

Part 2

- Navigate back to GitLab01 in your terminal
- Clone [this GitHub repository](https://github.com/Zipcoder/gh-user-names.git)
- Change into the new directory that you just cloned and create a file containing your GitHub user name; the file name should be your name.
- add and commit the file you changed, then try to push your changes to the repository you cloned from (you will find that you don't have permission to push to the repository).

Links

- "this GitHub repository" - https://github.com/Zipcoder/gh-user-names.git

CD-2 Git Lab Level 2

Objectives

1. In this lab students will practice forking, branching, and merging branches.
2. After this lab students should be comfortable creating feature branches and working on projects forked from other git repositories

Overview

This lab will give you a chance to fork a project on GitHub. Forking is the act of making a copy of a project on your own account. GitHub keeps track of which project is the original and how many times it has been forked. You will then practice making changes to your copy of the project using feature branches to hold your work, and eventually you will propose that your changes be included in the original project by submitting a Pull Request on GitHub.

Unit Test

Your work will be checked to confirm it includes the files created during this lab. We will also check the logs of your git repository to confirm that you created branches to house your changes while they were in progress.

Instructions

Part 1 Forking and Feature Branches

- Visit this github repo and click the "Fork" button to create a copy of it in your account
- Clone your new repo into a local directory on your machine.
- In your local repo, create a branch called `favorite-foods`
- Check out the new branch you created
- Create a new directory. The directory name should match your name.
- In the directory with your name, create a file called "FavoriteFoods.md"
- In FavoriteFoods.md, list some of your favorite foods. You can look at the example files provided for the instructors for examples and Markdown Basics for help with formatting your file.
- Use the git commands you practiced in the last lab to add your file to staging and commit it to your local repository.

Part 2 Working on Another Feature

You should probably have a short bio ready before posting factoids about things like favorite foods. Let's take care of that now.

- Check out your `master` branch again, and create a new branch from it called `bio`
- Check out the `bio` branch and create a file in your directory called "Bio.md". Notice that the file you committed to your `favorite-foods` branch isn't there (You may also need to recreate the directory).
- Write a short bio about yourself in the new file, add and commit it to the current branch.

Part 3: Merging

Now that you have changes in two different branches, it's time to practice merging. It's important to note that these changes **do not** overlap (you didn't change the same file in two different branches) -- the merge process gets a bit more complicated when merging two branches that change the same file.

- Check out your `master` branch. We'll merge our changes into this branch from the other two.
- Use the command `git merge branchname` to merge changes from `bio` onto the current branch (master) -- notice that the console message mentions "Fast Forward", this means that `master` had no changes since `bio` was created, so all commits in `bio` could be added to `master` by simply fast-forwarding to the latest commit.
- Try running the same merge command again (same branchname as before) and notice the new message:

"Already up-to-date" -- git is letting you know that there are no changes to merge (this will also happen if you try to merge a branch with no changes into a branch that is ahead of it.)

- Now that you've merged `bio` into `master`, look at your directory to confirm that the updated bio is there. You can also view the list of commits by using the `git log` command
- It's time to merge your list of favorite foods into `master`. If you want, you can check out your `favorite-foods` branch and make some more changes (remember not to change `Bio.md` or you will create a conflict -- we'll talk about those in the next lab).
- Once you've made all the changes you want (and committed them on your `favorite-foods` branch) check out your `master` branch and use the merge command from earlier to merge `favorite-foods` onto master. This time you will see an editor pop up (Most likely VIM). The default commit message is fine for now, so go ahead an save the message and exit the editor (`:wq` in VIM -- w for write and q for quit; basic VIM commands and other editor options are covered in the BASH labs)
- Note that this time the merge method used is called "recursive" -- git had to actually compare the changes made on each branch to figure out how to merge them. Check the git log to see that all of your commits are now present on `master`
- Now that you are finished with both feature branches (`bio` and `favorite-foods`) it's safe to delete them using the `git branch -d branchname` command.

Part 4 Pull Requests

You're all set with your changes now. It's time to push them to the GitHub copy of your repository (remember, all of the changes so far have only been made to your local repository on your laptop -- if you want to share them with others you have to update the repository on GitHub to contain the same changes)

- First, push your changes to your remote repository on GitHub. By default when you clone a repository git sets up a remote called `origin` but you can double check that with the `git remote` command. The syntax for push is `git push remotename branchname`
- Now navigate to your GitHub repository in your browser and click the "New Pull Request" button.
- If you've followed the instructions so far, you should see a comparison of your changed files to the current state of the original repository you forked. Make sure the changes you made are all there and that nothing else has accidentally changed and click "Create Pull Request"
- Finish filling out the pull request with a brief description of your changes and submit it for review.

Additional Resources

If you're confused or you want more practice, try one of these resources:

- Github guide to Forking Projects
- GitHub Pull Request Help

Links

- "this GitHub repo" - https://github.com/Zipcoder/gh-user-names.git
- "Markdown Basics" - https://help.github.com/articles/markdown-basics/
- "Forking Projects" - https://guides.github.com/activities/forking/index.html
- "Pull Request Help" - https://help.github.com/articles/using-pull-requests/